

**Listing of Claims**

1. (Currently amended) A method for efficiently dispatching threads awaiting messages in a multi-threaded communication library comprising:

preassigning threads to messages to be received;  
putting to sleep, those threads whose assigned messages have not been received;  
upon receipt of a message, awakening its preassigned thread; and  
executing said awakened thread[,]  
which processes [thereby processing] the received message and tests to see if any threads are ready to run.

2. (Original) The method of claim 1 wherein the selection of the thread to be dispatched is based on its priority as set when the thread is put to sleep.

3. (Original) The method of claim 1 wherein said preassigning threads step comprises:  
creating a thread-specific structure for each thread, each thread-specific structure having a ready flag and a condition variable unique to its preassigned thread;  
creating a handle for each message to be received; and  
having a thread invoke message passing logic for a particular handle, thereby associating the thread and the message.

4. (Original) The method of claim 3 wherein said putting to sleep step comprises:  
enqueueing for a received message, a preassigned thread-specific structure into a first queue;  
writing into said handle associated with the message received, an identification of said thread-specific structure enqueued for the received message, and  
placing said thread-specific structure for the received message in the WAIT condition.

5. (Original) The method of claim 4 wherein said awakening step comprises;  
    completing said received message;  
    changing the condition of the thread-specific structure for the completed received structure to the READY condition; and  
    dequeuing with a queue manager, the next thread-specific structure in said first queue in the READY condition and sending its thread a thread awakening condition signal.
6. (Original) The method of claim 5 further comprising;  
    allocating in said preassigning step, buffer space for storing messages to be received; and  
    in said putting to sleep step, identifying in said handle the buffer in which the message associated with the handle is to be stored when it is received.
7. (Original) The method of claim 6 wherein said completing said received message comprises storing said received message in the buffer identified in the associated handle for the received message.
8. (Original) The method of claim 5 wherein said queue manager dequeues the next thread-specific structure using a First-In-First-Out policy.
9. (Original) The method of claim 5 wherein said queue manager dequeues the next thread-specific structure using a Last-In-First-Out policy.
10. (Original) The method of claim 5 wherein said queue manager dequeues the next thread-specific structure based on a priority value contained in said structure.

11. (Original) The method of claim 5 further comprising obtaining a lock for the handle associated with said received message such that the awakened thread may process only the received message.

12. (Original) The method of claim 11 further comprising releasing said lock after said awakened thread has processed said received message such that said awakened thread may continue with other work.

13. (Currently amended) A computer program product comprising a computer useable medium having computer readable program code means therein for efficiently dispatching threads awaiting messages in a multi-threaded communication library, said computer readable program code means in said computer program product comprising:

computer readable program code means for preassigning threads to messages to be received;

computer readable program code means for putting to sleep, those threads whose assigned messages have not been received;

computer readable program code means for, upon receipt of a message, awakening its preassigned thread; and

computer readable program code means for executing said awakened thread, [thereby] processing the received message and testing to see if any threads are ready to run.

14. (Original) The computer program product of claim 13 wherein the selection of the thread to be dispatched is based on its priority as set when the thread is put to sleep.

15. (Original) The computer program product of claim 13 wherein said computer readable program code means for preassigning threads comprises:

computer readable program code means for creating a thread-specific structure for each thread, each thread-specific structure having a ready flag and a condition variable unique to its preassigned thread;

computer readable program code means for creating a handle for each message to be received; and

computer readable program code means for having a thread invoke message passing logic for a particular handle, thereby associating the thread and the message.

16. (Original) The computer program product of claim 15 wherein said computer readable program code means for putting to sleep comprises:

computer readable program code means for enqueueing for a received message, a preassigned thread-specific structure into a first queue;

computer readable program code means for writing into said handle associated with the message received, an identification of said thread-specific structure enqueued for the received message, and

computer readable program code means for placing said thread-specific structure for the received message in the WAIT condition.

17. (Original) The computer program product of claim 16 wherein said computer readable program code means for awakening comprises;

computer readable program code means for completing said received message;

computer readable program code means for changing the condition of the thread-specific structure for the completed received structure to the READY condition; and

computer readable program code means for dequeuing with a queue manager, the next thread-specific structure in said first queue in the READY condition and sending its thread a thread awakening condition signal.

18. (Original) The computer program product of claim 17 further comprising;

computer readable program code means for allocating in said preassigning step, buffer space for storing messages to be received; and

said computer readable program code means for putting to sleep includes, computer readable program code means for identifying in said handle the buffer in which the message associated with the handle is to be stored when it is received.

19. (Original) The computer program product of claim 18 wherein said computer readable program code means for completing said received message comprises computer readable program code means for storing said received message in the buffer identified in the associated handle for the received message.

20. (Original) The computer program product of claim 17 wherein said queue manager includes computer readable program code means for dequeuing the next thread-specific structure using a First-In-First-Out policy.

21. (Original) The computer program product of claim 17 wherein said queue manager includes computer readable program code means for dequeuing the next thread-specific structure using a Last-In-First-Out policy.

22. (Original) The computer program product of claim 17 wherein said queue manager includes computer readable program code means for dequeuing the next thread-specific structure based on a priority value contained in said structure.

23. (Original) The computer program product of claim 17 further comprising computer readable program code means for obtaining a lock for the handle associated with said received message such that the awakened thread may process only the received message.

24. (Original) The computer program product of claim 23 further comprising computer readable program code means for releasing said lock after said awakened thread has processed said received message such that said awakened thread may continue with other work.

25. (Currently amended) An apparatus for efficiently dispatching threads awaiting messages in a multi-threaded communication library comprising:

means for preassigning threads to messages to be received;

means for putting to sleep, those threads whose assigned messages have not been received;

means for, upon receipt of a message, awakening its preassigned thread; and

executing said awakened thread[,] which processes [thereby processing] the received message and tests to see if any threads are ready to run.

26. (Original) The apparatus of claim 25 wherein the selection of the thread to be dispatched is based on its priority as set when the thread is put to sleep.

27. (Original) The apparatus of claim 25 wherein said means for preassigning threads comprises:

means for creating a thread-specific structure for each thread, each thread-specific structure having a ready flag and a condition variable unique to its preassigned thread;

means for creating a handle for each message to be received; and

means for having a thread invoke message passing logic for a particular handle, thereby associating the thread and the message.

28. (Original) The apparatus of claim 27 wherein said means for putting to sleep comprises:

means for enqueueing for a received message, a preassigned thread-specific structure into a first queue;

means for writing into said handle associated with the message received, an identification of said thread-specific structure enqueued for the received message, and

means for placing said thread-specific structure for the received message in the WAIT condition.

29. (Original) The apparatus of claim 28 wherein said means for awakening comprises;

means for completing said received message;

means for changing the condition of the thread-specific structure for the completed received structure to the READY condition; and

means for dequeuing with a queue manager, the next thread-specific structure in said first queue in the READY condition and sending its thread a thread awakening condition signal.

30. (Original) The apparatus of claim 29 further comprising;

means for allocating in said preassigning step, buffer space for storing messages to be received; and

in said means for putting to sleep, means for identifying in said handle the buffer in which the message associated with the handle is to be stored when it is received.

31. (Original) The apparatus of claim 30 wherein said means for completing said received message comprises means for storing said received message in the buffer identified in the associated handle for the received message.

32. (Original) The apparatus of claim 29 wherein said queue manager includes means for dequeuing the next thread-specific structure using a First-In-First-Out policy.

33. (Original) The apparatus of claim 29 wherein said queue manager includes means for dequeuing the next thread-specific structure using a Last-In-First-Out policy.

34. (Original) The apparatus of claim 29 wherein said queue manager includes means for dequeuing the next thread-specific structure based on a priority value contained in said structure.

35. (Original) The apparatus of claim 29 further comprising means for obtaining a lock for the handle associated with said received message such that the awakened thread may process only the received message.

36. (Original) The apparatus of claim 35 further comprising means for releasing said lock after said awakened thread has processed said received message such that said awakened thread may continue with other work.

37. (Currently amended) An apparatus comprising:

- a data processing system;
- a multi-threaded communication library in said data processing system;
- a thread dispatcher in said data processing system for efficiently dispatching threads awaiting messages in said multi-threaded communication library;
- computer code which preassigns threads to messages to be received;
- computer code which puts to sleep those threads whose assigned messages have not been received;
- computer code which, upon receipt of a message, awakens its preassigned thread; and
- computer code which executes said awakened thread, thereby processing the received message and testing to see if any threads are ready to run.

38. (Original) The apparatus of claim 37 wherein the selection of the thread to be dispatched is based on its priority as set when the thread is put to sleep.

39. (Original) The apparatus of claim 37 wherein said computer code which preassigns threads comprises:

computer code which creates a thread-specific structure for each thread, each thread-specific structure having a ready flag and a condition variable unique to its preassigned thread;

computer code which creates a handle for each message to be received; and

computer code which causes a thread invoke message passing logic for a particular handle, thereby associating the thread and the message.

40. (Original) The apparatus of claim 39 wherein said computer code which puts to sleep comprises:

computer code which enqueues for a received message, a preassigned thread-specific structure into a first queue;

computer code which writes into said handle associated with the message received, an identification of said thread-specific structure enqueued for the received message, and

computer code which places said thread-specific structure for the received message in the WAIT condition.

41. (Original) The apparatus of claim 40 wherein said computer code which awakens comprises;

computer code which completes said received message;

computer code which changes the condition of the thread-specific structure for the completed received structure to the READY condition; and

computer code which dequeues with a queue manager, the next thread-specific structure in said first queue in the READY condition and sending its thread a thread awakening condition signal.

42. (Original) The apparatus of claim 41 further comprising;

in said computer code which preassigns, computer code which allocates buffer space for storing messages to be received; and

in said computer code which puts to sleep, computer code which identifies in said handle the buffer in which the message associated with the handle is to be stored when it is received.

43. (Original) The apparatus of claim 42 wherein said computer code which completes said received message comprises computer code which stores said received message in the buffer identified in the associated handle for the received message.

44. (Original) The apparatus of claim 41 wherein said queue manager includes computer code which dequeues the next thread-specific structure using a First-In-First-Out policy.

45. (Original) The apparatus of claim 41 wherein said queue manager includes computer code which dequeues the next thread-specific structure using a Last-In-First-Out policy.

46. (Original) The apparatus of claim 41 wherein said queue manager includes computer code which dequeues the next thread-specific structure based on a priority value contained in said structure.

47. (Original) The apparatus of claim 41 further comprising computer code which obtains a lock for the handle associated with said received message such that the awakened thread may process only the received message.

48. (Original) The apparatus of claim 47 further comprising computer code which releases said lock after said awakened thread has processed said received message such that said awakened thread may continue with other work.